



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/605,189	09/12/2003	Gregor P. Freund	VIV/0011.01	2188

28653 7590 04/10/2007
JOHN A. SMART
708 BLOSSOM HILL RD., #201
LOS GATOS, CA 95032

EXAMINER

HA, LEYNNA A

ART UNIT	PAPER NUMBER
----------	--------------

2135

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/10/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary

Application No.

10/605,189

Applicant(s)

FREUND, GREGOR P.

Examiner

LEYNNA T. HA

Art Unit

2135

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 16 January 2007.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-47 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-47 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
- ☐ Certified copies of the priority documents have been received.
 - ☐ Certified copies of the priority documents have been received in Application No. _____.
 - ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-47 is pending.
2. This is a Final rejection.

Claim Rejections - 35 USC § 102

The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

3. **Claims 1-47 are rejected under 35 U.S.C. 102(e) as being anticipated by Andrews (US 6,574,736).**

As per claim 1:

Andrews discloses a method for controlling interprocess communication, the method comprising:

defining rules indicating which system services a given application can invoke (**col.9, lines 34-38 and 49-55; Andrews disclose object oriented programming, programs are written as a collection of object class which**

each model real world or abstract items by combining data to represent the item's properties with methods (e.g. program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism, and inheritance (col.2, lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or module (col.6, lines 23-26). With this in mind, Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications. More than two roles can be associated, and the roles maybe from the same or different applications (col.4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col.4, lines 47-53). Thus, the rules refer to the roles of the applications in addition to being populated with users or the types of users able to access the applications (col.4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.) using interprocess communication to invoke said system services; (col.16, lines 16-23 and col.21, lines 36-53; the interprocess communication is the ability of a task or process to communicate with another in a multitasking operating system (col.1, lines 50-54 and col.2, lines 47-50). Hence, an object being executed to communicate via

signals (col.8, lines 1-15) with another to perform different functions (e.g. defining and regulating rules or transferring processes) or tasks being performed would read on the claimed interprocess communications (col.6, lines 34-40 and col.8, lines 43-51). Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads. When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user. Andrews further teaches the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col.5, lines 63-67). Thus, Andrews reads on the claimed using interprocess communication to invoke system services (col.19, lines 15-30).)

trapping an attempt by a particular application to invoke a particular system service; **(col.15, lines 20-22 and col.21, lines 35-37)**

identifying the particular application that is attempting to invoke the particular system service; and **(col.20, lines 15-16 and col.22, lines 13-18 and 30-32)**

based on identity of the particular application and on the rules indicating which system services a given application can invoke **(col.14, lines 56-59),**

Art Unit: 2135

blocking the attempt when the rules indicate that the particular application cannot invoke the particular system service. **(col.22, lines 1-9)**

As per claim 2: See col.15, lines 20-21; discussing the method of claim 1, wherein said trapping step includes intercepting operating system calls for invoking the particular system service.

As per claim 3: See col.15, lines 20-21; discussing the method of claim 1, wherein said trapping step includes intercepting local procedure calls for invoking the particular system service.

As per claim 4: See col.16, lines 65-67 and col.21, lines 35-37; discussing the method of claim 1, wherein said trapping step includes intercepting an attempt to open a communication channel to the particular system service.

As per claim 5: See col.16, lines 17-31; discussing the method of claim 1, wherein said trapping step includes rerouting an attempt to invoke the particular system service from a system dispatch table to an interprocess communication controller for determining whether to block the attempt based on the rules.

As per claim 6: See col.8, lines 56-64; discussing the method of claim 5, wherein said step of rerouting attempts to invoke the particular system service from a dispatch table to the interprocess communication controller includes replacing an original destination address in the system dispatch table with an

address of the interprocess communication controller.

As per claim 7: See col.8, lines 58-59; discussing the method of claim 6, further comprising the steps of: retaining the original destination address; and using the original destination address for invoking the particular system service if the interprocess communication controller determines not to block the attempt.

As per claim 8: See col.9, lines 34-36; discussing the method of claim 1, wherein the rules specifying which system services a given application can invoke are established based on user input.

As per claim 9: See col.14, line 53 – col.15, line 3; discussing the method of claim 1, wherein the step of blocking the attempt is based upon consulting a rules engine for determining whether the particular application can invoke the particular system service.

As per claim 10: See col.22, lines 1-9; discussing the method of claim 1, wherein the step of blocking the attempt includes obtaining user input as to whether the particular application can invoke the particular system service.

As per claim 11: See col.15, lines 20-30 and col.23, lines 7-14; discussing the method of claim 10, wherein said step of obtaining user input as to whether the particular application can invoke the particular system service includes the substeps of: providing information to the user about the particular application that is attempting to invoke the particular system service; and receiving user input as to whether the particular application should be blocked

from invoking the particular system service.

As per claim 12: See col.2, lines 47-49; discussing the computer-readable medium having computer-executable instructions for performing the method of claim 1.

As per claim 13: See col.1, lines 10-30; discussing downloading a set of computer-executable instructions for performing the method of claim 1.

As per claim 14:

Andrews discloses in a computer system, a method for regulating communications between processes, the method comprising:

defining a policy specifying whether one process may use interprocess communication to communicate with another process; **(col.9, lines 34-38 and 49-55; Andrews disclose object oriented programming, programs are written as a collection of object class which each model real world or abstract items by combining data to represent the item's properties with methods (e.g. program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism, and inheritance (col.2, lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or module (col.6, lines 23-26). With this in mind, Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications. More**

than two roles can be associated, and the roles maybe from the same or different applications (col.4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col.4, lines 47-53). Thus, the rules refer to the roles of the applications in addition to being populated with users or the types of users able to access the applications (col.4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.)

intercepting an attempt by a first process to communicate with a second process; **(col.15, lines 20-22 and col.21, lines 35-37)**

identifying the first process that is attempting to communicate with the second process; **(col.20, lines 33-57 and col.22, lines 1-3)**

identifying the second process; **(col.13, lines 2-3 and col.21, lines 49-67)**

based on said policy, determining whether the first process may communicate with the second process; and **(col.14, lines 56-59 and col.22, lines 13-18 and 30-32)**

allowing **(col.22, lines 25-57)** the first process to communicate with the second process if said policy indicates that the first process may communicate with the second process. **(col.16, lines 16-23 and col.21, lines 36-53; the interprocess communication is the ability of a task or process to**

communicate with another in a multitasking operating system (col.1, lines 50-54 and col.2, lines 47-50). Hence, an object being executed to communicate via signals (col.8, lines 1-15) with another to perform different functions (e.g. defining and regulating rules or transferring processes) or tasks being performed would read on the claimed interprocess communications (col.6, lines 34-40 and col.8, lines 43-51). Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads. When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user. Andrews further teaches the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col.5, lines 63-67). Thus, Andrews reads on the claimed using interprocess communication to invoke system services (col.19, lines 15-30).)

As per claim 15: See col.2, lines 55-57; discussing the method of claim 14, wherein the first process comprises an instance of an application program.

As per claim 16: See col.2, lines 55-57; discussing the method of claim 14, wherein the second process comprises a system service.

As per claim 17: See col.15, lines 20-22 and col.21, lines 35-37;

Art Unit: 2135

discussing the method of claim 14, wherein said intercepting step includes intercepting operating system calls made by the first process to attempt to communicate with the second process.

As per claim 18: See col.15, lines 20-21; discussing the method of claim 14, wherein said intercepting step includes detecting local procedure calls.

As per claim 19: See col.16, lines 65-67 and col.21, lines 35-37; discussing the method of claim 14, wherein said intercepting step includes detecting an attempt by the first process to open a communication channel to the second process.

As per claim 20: See col.col.15, lines 34-40 and col.16, lines 17-31; discussing the method of claim 14, wherein said intercepting step includes rerouting attempts by the first process to communicate with the second process from a system dispatch table to an interprocess communication controller.

As per claim 21: See col.20, lines 33-57 and col.22, lines 1-3; discussing the method of claim 14, wherein said step of identifying the second process includes evaluating parameters of the attempt made by the first process to communicate with the second process.

As per claim 22: See col.8, lines 55-66 and col.9, lines 50-55; discussing the method of claim 14, wherein said policy specifies particular processes to be protected from communications made by other processes.

As per claim 23: See col.1, lines 42-52 and col.22, lines 25-57; discussing

Art Unit: 2135

the method of claim 14, further comprising: providing for a process to be registered in order to be protected from communications made by other processes; and determining whether to allow the first process to communicate with the second process based, at least in part, upon determining whether the second process is registered.

As per claim 24: See col.7, lines 35-47; discussing the method of claim 23, wherein said determining step is based, at least in part, on the type of communication the first process is attempting with the second process.

As per claim 25:

Andrews discloses a method for controlling interprocess communications from one application to another, the method comprising:

registering (**col.12, lines 35-67**) a first application to be protected from interprocess communications of other applications; (**col.9, lines 34-38 and 49-55; Andrews disclose object oriented programming, programs are written as a collection of object class which each model real world or abstract items by combining data to represent the item's properties with methods (e.g. program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism, and inheritance (col.2, lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or module (col.6, lines 23-26). With this in mind,**

Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications. More than two roles can be associated, and the roles maybe from the same or different applications (col.4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col.4, lines 47-53). Thus, the rules refer to the roles of the applications in addition to being populated with users or the types of users able to access the applications (col.4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.)

detecting an attempt to access the first application using interprocess communication; **(col.20, lines 33-57 and col.22, lines 1-3)**

identifying a second application that is attempting to access the first application using interprocess communication; and **(col.13, lines 2-3 and col.21, lines 49-67)**

rerouting the attempt to access the first application through an interprocess communication controller that determines whether to allow **(col.6, lines 42-43 and col.22, lines 25-57)** the attempt, based on rules indicating whether the second application may access the first application using interprocess communication. **(col.16, lines 16-23 and col.21, lines 36-53; the interprocess communication is the ability of a task or process to**

communicate with another in a multitasking operating system (col.1, lines 50-54 and col.2, lines 47-50). Hence, an object being executed to communicate via signals (col.8, lines 1-15) with another to perform different functions (e.g. defining and regulating rules or transferring processes) or tasks being performed would read on the claimed interprocess communications (col.6, lines 34-40 and col.8, lines 43-51). Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads. When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user. Andrews further teaches the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col.5, lines 63-67). Thus, Andrews reads on the claimed using interprocess communication to invoke system services (col.19, lines 15-30).)

As per claim 26: See col.16 lines 1-8; discussing the method of claim 25, wherein said registering step includes supplying rules specifying particular communications from which the first application is to be protected.

As per claim 27: See col.15, lines 35-41 and col.22, lines 25-57; discussing the method of claim 26, wherein the interprocess communication

controller determines whether to allow the attempt based, at least in part, upon the rules specifying particular communications from which the first application is to be protected.

As per claim 28: See col.15, lines 20-22 and col.21, lines 35-37;

discussing the method of claim 25, wherein said detecting step includes intercepting operating system calls for accessing the first application.

As per claim 29: See col.10, lines 55-56; discussing the method of claim 25, wherein said detecting step includes detecting a graphical device interface (GDI) message sent to the first application.

As per claim 30: See col.20, lines 33-57 and col.22, lines 1-3; discussing the method of claim 29, wherein said identifying step includes evaluating parameters of the message sent to the first application.

As per claim 31: See col.7, lines 23-24; discussing the method of claim 25, wherein said detecting step includes detecting an attempt to send keystroke data to a window of the first application.

As per claim 32: See col.7, lines 23-25; discussing the method of claim 25, wherein said detecting step includes detecting an attempt to send mouse movement data to a window of the first application.

As per claim 33: See col.15, lines 35-41 and col.22, lines 25-57;

discussing the method of claim 25, wherein said rerouting step includes rerouting the attempt to access the first application from a system dispatch table to the interprocess communication controller.

As per claim 34: See col.16, line 65-col.17, line 5; discussing the method of claim 25, wherein said rules indicating whether the second application may access the first application includes rules indicating particular types of communications which are allowed.

As per claim 35: See col.15, lines 35-41 and col.22, lines 25-57; discussing the method of claim 25, further comprising: if the interprocess communication controller allows the attempt to access the first application, routing the attempt to the first application.

As per claim 36:

Andrews discloses a system for regulating interprocess communication between applications, the system comprising:

a policy specifying applications (**col.20, lines 33-57 and col.22, lines 1-3**) that are permitted to communicate with a first application using interprocess communication; (**col.9, lines 34-38 and 49-55; Andrews disclose object oriented programming, programs are written as a collection of object class which each model real world or abstract items by combining data to represent the item's properties with methods (e.g. program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism, and inheritance (col.2, lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or**

module (col.6, lines 23-26). With this in mind, Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications. More than two roles can be associated, and the roles maybe from the same or different applications (col.4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col.4, lines 47-53). Thus, the rules refer to the roles of the applications in addition to being populated with users or the types of users able to access the applications (col.4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.)

a module for detecting a second application attempting to communicate with the first application using interprocess communication; and **(col.13, lines 2-3 and col.21, lines 49-67)**

an interprocess communication controller for identifying the second application attempting to communicate with the first application and determining whether to permit **(col.6, lines 42-43 and col.22, lines 25-57)** the communication based upon the identification of the second application and the policy specifying applications permitted to communicate with the first application. **(col.16, lines 16-23 and col.21, lines 36-53; the interprocess communication is the ability of a task or process to communicate with another in a multitasking operating system (col.1, lines 50-54 and col.2,**

lines 47-50). Hence, an object being executed to communicate via signals (col.8, lines 1-15) with another to perform different functions (e.g. defining and regulating rules or transferring processes) or tasks being performed would read on the claimed interprocess communications (col.6, lines 34-40 and col.8, lines 43-51). Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads. When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user. Andrews further teaches the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col.5, lines 63-67). Thus, Andrews reads on the claimed using interprocess communication to invoke system services (col.19, lines 15-30).)

As per claim 37: See col.16 lines 1-8; scussing the system of claim 36, wherein said policy includes rules indicating particular types of communications which are permitted.

As per claim 38: See col.15, lines 35-41 and col.22, lines 25-57; discussing the system of claim 36, further comprising: a rules engine for specifying applications that are permitted to communicate with the first

application using interprocess communication.

As per claim 39: See col.12, lines 35-67; discussing the system of claim 36, further comprising: a registration module for establishing said policy.

As per claim 40: See col.12, lines 35-67; discussing the system of claim 39, wherein said registration module provides for identifying applications to be governed by said policy.

As per claim 41: See col.16, line 65-col.17, line 5; discussing the system of claim 36, wherein said module for detecting a second application detects an operating system call to open a communication channel to the first application.

As per claim 42: See col.10, lines 55-56; discussing the system of claim 36, wherein said module for detecting a second application detects a graphical device interface (GDI) message sent to the first application.

As per claim 43: See col.13, lines 2-3 and col.21, lines 49-67; discussing the system of claim 36, wherein said module for detecting a second application detects a local procedure call attempting to access the first application.

As per claim 44: See col.15, lines 35-41 and col.22, lines 25-57; discussing the system of claim 36, wherein said module for detecting a second application redirects attempts to communicate with the first application to the interprocess communication controller.

As per claim 45: See col.15, lines 35-41 and col.22, lines 25-57; discussing the system of claim 36, wherein said module for detecting a second application reroutes the attempt to communicate with the first application from

a dispatch table to the interprocess communication controller.

As per claim 46: See col.15, lines 35-41 and col.22, lines 25-57;

discussing the system of claim 36, wherein said interprocess communication controller determines whether to permit the communication based, at least in part, upon evaluating parameters of the attempt made by the second application to communicate with the first application.

As per claim 47: See col.16, line 65-col.17, line 5 and col.22, lines 25-57;

discussing the system of claim 36, wherein said interprocess communication controller determines whether to permit the communication based upon obtaining user input as to whether to permit the second application to communicate with the first application.

Response to Arguments

4. Applicant's arguments filed 1/16/2007 have been fully considered but they are not persuasive.

Andrews disclose object oriented programming, programs are written as a collection of object class which each model real world or abstract items by combining data to represent the item's properties with methods (e.g. program functions or procedures) to represent the item's functionality. An object is an instance of a programmer defined type referred to as a class, which exhibits the characteristics of data encapsulation, polymorphism, and inheritance (col.2,

lines 1-10). The term application can also broadly give in light as a program, object, software, routine, or module (col.6, lines 23-26).

Further, Examiner traverses the arguments throughout pages 12-15, indicating that Andrews's focus is on controlling user access because if in deed Andrews discloses user access is merely additional features to his invention. All the recitation throughout the rejection points to the claimed invention that focuses on the applications and its roles. Plus, Andrew discloses the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications (col.5, lines 63-67). More than two roles can be associated, and the roles maybe from the same or different applications (col.4, lines 27-28). In addition, the security framework including role definitions, to determine whether access to a component's functionality is permitted where the framework blocks calls to objects if access is not permitted (col.4, lines 47-53). With this in mind, Andrews discloses application role is a role defined of the application to define access privileges to processing services to the applications in addition to user roles (col.9, lines 33-42). Hence, Andrews does not only include rules for applications but rules for users where this does not mean that Andrews only discloses user rules as applicant alleges. In addition, the user is not the focus as having rules, it is in the application in which users perform functions such as "shippers" and "return staff" roles are defined (col.4, lines 9-15). Thus, the rules refer to the roles of the applications

in addition to being populated with users or the types of users able to access the applications (col.4, lines 25-35). Thus, Andrews reads on the claimed defining rules for the applications.

The argument on page 14, regarding Andrews does not discuss interprocess communication between different processes is traversed because column 9 (as pointed out by applicant) is just an explanation about the difference between the user roles and an application roles. This only proves against applicant alleging that the prior art focuses only on user access. Applicant's invention broadly claims the method for controlling interprocess communication. The claimed interprocess communication can broadly interpret as the ability of a task or process to communicate with another in a multitasking operating system (col.1, lines 50-54 and col.2, lines 47-50). In essence, the claim merely requires controlling interprocess communication between two applications. The following are examples taught by Andrews to read on the claimed invention. Andrews discloses launching or executing a program where the operating system associates the user id with the process in which the program is run and with the process' threads. When a thread executing on the user's behalf then accesses a system resource, the operating system performs an authorization check to verify the user (col.1, lines 50-55). Andrew discloses the present invention is directed toward a method and system for providing an object execution environment with a security framework providing automatic security services for composable applications

(col.5, lines 63-67). Andrews discloses the invention is practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through communications network (col.6, lines 34-40). Andrews discusses the client object attempts to access the functionality of the server object and that the wrapper invokes various security function calls to the operating system and accesses a catalog to determine whether to relay the call to the server object (col.15, lines 33-41). Andrews also discloses various objects execute in a process on a computer where the objects 770, 772, 774 and members of one application and are accordingly placed in object context A; the objects 790 and 792 are members of another application and accordingly are placed in object context B. When an object in one application attempts to access the functionality of an object in the other application (col.16, lines 16-23). These various recitations suggest one or more applications communicating to another application if there is communication from one computer to another computer. Therefore, Andrews reads on the claimed defining rules indicating which system services a given application can invoke using interprocess communication to invoke system services.

Examiner traverses the argument on page 15 regarding Andrews discloses the claimed trapping where this intercepting is specific user-based intercepting. Examiner is unclear how or what exactly applicant means when alleging Andrews is specific to user-based intercepting because Andrews clearly shows in the application is invoked using the interprocess communication

method and that the objects are performing the intercepting (col.15, lines 20-40). Again, an object refers to the claimed application that is being invoked or executed. When Andrews uses the term “client object” is merely referring to an object that belongs to the user/client side or system. Andrews discloses the security framework monitors calls to the objects by intercepting calls to the components with a wrapper mechanism where FIG.13 shows two objects 706 and 708 as discussed in the previous paragraph. The objects are sometimes referred to as a client object 706 and a server object 708 since the object accesses functionality of (i.e. is served by) the object 708 (col.15, lines 20-40). Andrews also discusses client object access the functionality of the server object through a pair of wrappers and transparently interposed between the two objects during instantiation of the server object wherein the two objects are different applications (col.21, lines 31-53). Andrews also discloses various objects execute in a process on a computer where the objects 770, 772, 774 and members of one application and are accordingly placed in object context A; the objects 790 and 792 are members of another application and accordingly are placed in object context B. When an object in one application attempts to access the functionality of an object in the other application (col.16, lines 16-23). Andrews does include users and what applications the users are permitted to access but mainly concerns with the applications that are communicating to each other and its (application) roles to define access privileges to processing services to the application when one object wishes to

access the another object (col.9, lines 35-38 and col.19, lines 15-30). This does not teach away from applicant's invention, it is merely additional features of Andrews' invention.

Conclusion

5. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

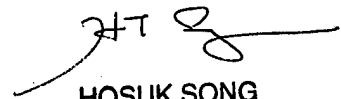
A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to LEYNNA T. HA whose telephone number is (571) 272-3851. The examiner can normally be reached on Monday - Thursday (7:00 - 5:00PM).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kim Vu can be reached on (571) 272-3859. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

LHa


HOSUK SONG
PRIMARY EXAMINER